Chapter 1

BETTER SOLUTIONS FASTER: SOFT EVOLUTION OF ROBUST REGRESSION MODELS IN PARETO GENETIC PROGRAMMING

Ekaterina Vladislavleva[1], Guido Smits[2] and Mark Kotanchek[3]
[1]*Tilburg University, Tilburg, the Netherlands;* [2]*Dow Benelux B.V., Terneuzen, the Netherlands;* [3]*Evolved-Analytics,LLC, Midland, MI, USA.*

Abstract "Better solutions faster" - is the reality of the industrial modeling world, now more than ever. Efficiency requirements, market pressures, and ever changing data force us to use symbolic regression via genetic programming (GP) in a highly automated fashion. This is why, we want our GP system to produce simple solutions of the highest possible quality with the lowest computational effort, and a high consistency in the results of independent GP runs.

In this chapter, we show that genetic programming with a focus on ranking in combination with goal softening are very powerful ways to improve the efficiency and effectiveness of the evolutionary search. Our strategy consists of partial fitness evaluations of individuals on random subsets of the original data set, with a gradual increase in the subset size in consecutive generations. From a series of experiments performed on three test problems, we observed that those evolutions that started from the smallest subset sizes (10%) consistently led to results that are superior in terms of the goodness of fit, consistency between independent runs, and computational effort. Our experience indicates that solutions obtained using this approach are also less complex and more robust against over-fitting.

We find that the near-optimal strategy of allocating computational budget over a GP run is to evenly distribute it over all generations. This implies that initially, more individuals can be evaluated using small subset sizes, promoting better exploration. Exploitation becomes more important towards the end of the run, when all individuals are evaluated using the full data set with correspondingly smaller population sizes.

Keywords: partial evaluation, archiving, budget allocation, goal softening, ranking, robust solutions, nonlinear regression, ordinal optimization,

1. Introduction

Symbolic Regression via genetic programming is a stochastic iterative search technique that automatically generates a rich set of data-driven symbolic models expressing the response variable of interest as an analytical function of given input variables. Such genetic programming system exploits a population of symbolic models (=expressions =individuals = formulae) to cover multiple points of the search space simultaneously. At each iteration step the population of alternatives is evaluated, and a subset of 'best' solutions discovered so far is selected to be optimized and produce the next, hopefully better generation.

The search space of a real-life symbolic regression problem is tremendously huge. The multidimensionality, noise, and inaccuracy of data, the nature-driven complexity of underlying relationships all contribute to make the search inherently difficult. The lack of any significant structure makes the navigation through this search space intrinsically hard.

Both increasing the size of the population and the length of the GP run will escalate computational requirements. Besides, the process can still suffer from bloat and premature convergence phenomena. The problem of bloat (Langdon and Poli, 2002), i.e. unnecessary growth of size of the solutions with no considerable improvements in fitness, is diminished by a Pareto-centric selection strategy (Smits and Kotanchek, 2004), (Zitzler and Thiele, 1998), (Laumanns et al., 2002), when both the goodness of fit and the expressional complexity of a model are optimized simultaneously.

The problem of premature convergence can be avoided by maintaining the diversity of the population. In our implementation of ParetoGP we use cascades to re-initialize the population completely on a periodical basis while maintaining an archive of solutions, lying near the Pareto front of model complexity and accuracy, see Section 2 and (Smits and Kotanchek, 2004; Kotanchek et al., 2006)).

Even with these modifications the majority of the computational effort is spent on fitness evaluation. One of the natural ways around this problem is to use subsets of data for fitness evaluations. Several groups have tried to address the issue of using parts of data for fitness evaluations without a loss in quality of solutions. Gathercole et al, (Gathercole and Ross, 1994), analyzed various selection schemes for partial fitness evaluation using subsets of data for a number of GP classification problems. This approach was later refined by Teller and Andre, (Teller and Andre, 1997), also for classification problems. Zhang and Cho, (Zhang and Cho, 1998), applied incremental subset selection to the evolution of collective behaviors for multiple robotic agents. We found no references that applied these principles to regression type problems.

While these subset selection schemes were mainly heuristic in nature, there exists a beautiful and simple theory, that, when combined with GP, has the

potential to provide a strong theoretical foundation to these approaches. This theory of Ordinal Optimization (OOpt) was developed by Yu-Chi Ho et. al (Ho, 2000). The ideas of Ordinal Optimization and more specifically of goal softening and ranking, not only fully concur with our thoughts and beliefs in evolutionary search, but couch them in strong language of mathematics and common sense.

The objective of this paper is threefold. At first, we would like to further extend and analyse the concept of ordinal ParetoGP. At second, we would like to introduce some additional goal softening to further reduce the cpu-budget while effectively keeping the same quality of the results. At third, we examine the effect of allocating more cpu-budget to exploration without changing the total budget for the total run.

The chapter is organized as follows. In Section 2 we briefly describe our Pareto-based genetic programming system that is used as a reference. In Section 3 we quickly summarize the main ideas of ordinal optimization. In Section 4 we analyse the effects of goal softening introduced by the use of subsets of the data to evaluate individuals in a GP population. Finally, in Sections 5 and 6 we describe and analyse a series of experiments leading to a set of robust settings for Ordinal ParetoGP.

2. Our Pareto-based Genetic Programming System

We have been working with a particular implementation of a genetic programming system, called ParetoGP. The main features that make it different from many other GP systems is an explicit consideration of multiple objectives to guide the search and the use of an archive of promising equations that is maintained during a run (see Table 1-1).

A foundation of every evolutionary system consists of granting higher propagation rights to *better* individuals. A classical regression via GP system uses one objective to decide on the quality of individuals. Since the main purpose of any regression system is to construct an accurate approximation of the observed output as a function of given inputs, accuracy (prediction error) of a GP individual is usually used as a performance measure. We believe that adequacy of a GP individual is of the same importance as the accuracy of its prediction. Therefore, for performance evaluation, we never use the prediction error alone, but always in a combination with a complexity measure.

In all experiments of this chapter only one fitness measure and one complexity measure were used. Fitness was determined as a normalized sum of squared errors between predicted output of a model py and an observed output y:

$$NMSE(y, py) = \frac{1 - MSE(y, py)}{1 + MSE(y, py)} \tag{1.1}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(rscale(y)i - rscale(py)_i)^2 \tag{1.2}$$

$$rscale(z) = \frac{z - \min(z)}{\max(z) - \min(z)} \tag{1.3}$$

The complexity measure of a symbolic model, expressed as a tree-structure, was determined as a sum of nodes in all subtrees of the given tree. We call it expressional complexity of a tree-based GP individual, (Smits et al., 2005). A thorough analysis of the properties of this measure is given in a recent paper of Maarten Keijzer and James Foster, who called it a visitation length, (Keijzer and Foster, 2007).

If two competing objectives are used for evaluating the performance of individuals, the relation of dominance needs to be defined for these individuals in the performance (objective) space. If small values of both objectives are preferred over bigger values (we are minimizing model error and model complexity), then an individual (an alternative) A with objective values (f_1, c_1) is said to dominate an individual B with objective values (f_2, c_2), if $f_1 \leq f_2$, $c_1 \leq c_2$, and either $f_1 < f_2$, or $c_1 < c_2$. In other words, A dominates B, if A is not worse than B in both objectives, and A is strictly better than B in at least one objective.

A set of individuals, non-dominated by any other individuals, forms a set of optimal trade-offs in the given objective space, and is called the Pareto front. A Pareto front in accuracy-complexity space refers to a set of best GP individuals, which should be granted the most propagation rights.

In the current implementation of ParetoGP the archive, i.e. the set of best individuals obtained by a given step of evolution has a predetermined maximum size. We define and update it at every generation with individuals that lie at the Pareto front of the combined populations of the new generation as well as the archive of the previous generation (see Figure 1-1 and Table 1-1).

To prevent inbreeding we re-initialize a population every k generations. By doing this we segment an evolutionary run into a series of *cascades* of k generations each, see (Smits and Kotanchek, 2004). Within a cascade the population is allowed to crossbreed with the archive to generate the next population. The archive is maintained during the entire run. For this reason, good solutions reappear in a population very quickly again despite re-initializations.

The performance measure that we typically use for monitor the progress of a GP run is either the fitness of the best individual in the archive or the percentage of area under the Pareto front defined by the fitness (model error) and the complexity. The latter performance metric is more robust and also reflects the balance between fitness and complexity we try to minimize (see (Smits and Vladislavleva, 2006)). In all experiments of this chapter we used

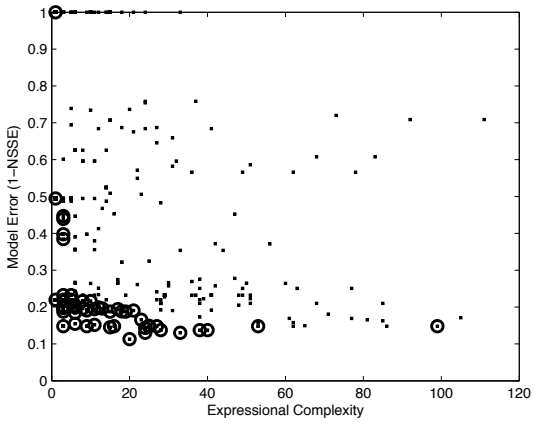


Figure 1-1. **Initializing an archive.** Black dots represent a population of 100 tree-based equations, plotted in the performance space of expressional complexity and accuracy (prediction error). Smaller values of both objectives are preferred over bigger ones. Circles represent the 50 'best' individuals obtained via a non-dominated sorting procedure. These individuals are the 50 'closest' to the Pareto front and will form the archive at the first generation.

Table 1-1. Our ParetoGP algorithm. The function $Generate$ produces $Population(t)$, from $Archive(t-1)$ and $Population(t-1)$ by means of structure changing operators. Best representatives of this set are used by the function $UpdateArchive$ to optimize the archive $Archive(t-1)$. When the iteration process is terminated, the set of solutions is determined by $Archive(t_{last})$, where t_{last} is the last iteration step. Function $UpdateArchive$ uses the union of the old archive and the new population to select a fixed number of models located at the Pareto front in complexity vs. fitness space.

```
t=0;     % generation count
% Initialize Archive
Archive(0)= ∅;
% Initialize population
Population(0)= ∅;
while search is not terminated
  t=t+1;
  % Generate new population from archive models at
  % the previous step
  Population(t) = Generate(Archive(t-1),Population(t-1));
  %Update Archive
  Archive(t) = UpdateArchive(Archive(t-1), Population(t));
end
% Return solutions of a GP run
Archive(t)
```

the area under the Pareto front of the archive at the last generation as a measure for *a posteriori* analysis of the effectiveness of a GP run.

3. Goal Softening and Ranking

The concept of Ordinal Optimization is described extensively in (Ho, 2000). This section is a summary of the main ideas of (Ho, 2000), (Ho et al., 2000), and (Lau and Ho, 1997). The theory rests on two basic tenets:

- It is easier to find a good enough solution with high confidence than the best solution for sure. Such *goal softening* helps to smooth and direct the search.
- It is easier to determine Order than Value, or, in other words, it is easier to determine whether $A > B$ than to determine A and B exactly.

When solving *hard* problems by computationally expensive (e.g. evolutionary) search-based methods, we argue that quickly narrowing down the search for an optimum to a 'good enough' subset of the search space is more important than accurate estimation of performance of potential solutions during the search.

Goal Softening. In real-life problems the true optimum is often unattainable and the compromise is made for 'good enough' solutions.

This substitution of:

- the best solution by a good enough subset of solutions,
- being sure by being confident with high probability,
- getting a closed form solution by an approximate solution, and
- making rational decision by using heuristics,

are all examples of softening the optimization goals.

Order vs. Value. Let us assume we have to quickly select the fastest runner of two. Instead of monitoring them separately for weeks and measuring the speed on various distances, we will let them compete with each other, and will choose the one who arrives at the finish first. In contrast, it is much easier to determine whether $A > B$, i.e. to determine the order, than to exactly estimate the difference between A and B under multiple conditions, i.e. to examine the value. Thus, although there is less certainty associated with the decision based on the ordinal approach, it is obtained at a much lower cost compared with the approach based on exhaustive evaluations.

4. Discussion: How to get better solutions faster by goal softening and emphasis on ranking?

Ordinal ParetoGP - better solutions with about the same effort

Considering the vastness of the search space we can improve the effectiveness of our evolutionary search procedure in either of two ways. One way is to try to get solutions of a similar quality at a lower computational cost. A second way is to try to get solutions of a better quality at the same or similar computational cost. We question whether we need exhaustive fitness evaluations to evolve solutions of a similar quality. If fitness evaluation can be 'softened', to what extent can we then decrease the size of the subsets of the data to perform this evaluation? We also wonder, what can be gained in terms of system performance and the cpu-budget, by evaluating more potential solutions at a lower cost and by gradually improving the fidelity of the fitness evaluations in the course of an evolution.

In symbolic regression via GP the bulk of the computational effort is spent predominantly on fitness evaluation of the individuals[1]. In many cases all available records of a given data set are used to determine the fitness of every individual. These finesses are then used to rank all models and then decide who gets the right to propagate and who does not. As we emphasized in a previous paper (Smits and Vladislavleva, 2006), it is not really critical what sort of selection system is being used - the essence is that all that is required is an ordering of the equations in terms of their performance (actually what we really need is not an ordering in terms of their current performance but an ordering in terms of their potential to generate even more fit offspring).

We also showed that the effectiveness and the reproducibility of our search can be improved considerably by introducing incomplete fitness evaluations. In the most successful scheme we used partial fitness evaluations on random subsets of the original data set while gradually increasing the subset size and decreasing the population size in consecutive generations. In the settings for this scheme we started with the subset size of 10% of the original size, and increased it to 100% during the first 200 of the total of 250 generations. At the same time the population size was decreased linearly from 1000 to 100 over the first 200 generations. The archive size was kept constant at 100 models at all times. These Ordinal ParetoGP runs outperformed standard ParetoGP runs of a 1000 generations each, both in terms of the final fitness as well as the consistency of 30 independent replicates.

[1] This holds for data sets of medium and large size, which are our main interest.

In the next section we explain the reasons for the improved performance we obtained in (Smits and Vladislavleva, 2006), by examining the influence of partial fitness evaluations on the ranking of individuals and the resulting change in the balance of exploration versus exploitation.

Better solutions because of better exploration. Instead of evaluating every model exhaustively using the full data set, we settle for using a subset of the available data to rank the models. The fact that we use a random subset to estimate the fitness of an individual introduces a certain level of noise in this estimate. By repeatedly selecting different subsets of a given size to determine the fitness of a given individual we can get an idea of what level of uncertainty we introduce by examining the resulting distributions. These distributions will obviously depend on the size of the subset we choose (smaller subsets will cause wider distributions) but will also depend on the individual itself (fitter individuals will have smaller distributions) (see Figure 1-2).

We have observed that the histograms of these distributions can be approximated by a normal distribution with the mean equal to the true fitness of the individual, and the standard deviation inversely proportional to the subset size and to the quality of the model [2] (see Figure 1-3).

When the true fitness distributions of two individuals overlap (see Figure 1-3), we can obviously make an error in taking the decision which is the best of the two. These 'mistakes' in the ranking are the first reason for enhanced exploration. A given individual can outrank another individual in the population but in addition that individual could also outperform and replace other individuals that are already in the archive. Since individuals that would be of lower fitness can now end up in the archive and contribute to creating offspring in the next generation, this increases the level of exploration in the search.

Another reason for better exploration is the fact that the use of subsets allows us to evaluate more models with the same computational budget. If we use subsets of a given size, sampled uniformly at random, they will be different for every generation. These will act as successive screens that every model will need to pass in order to survive long-term. The result is a regularizing effect and an increased robustness of the final models because there is less opportunity for pathologies or learning the noise that may be present in the data.

'About the same effort' because of archive re-evaluations. In the new approach we decided to keep the number of function evaluations per generation constant, therefore, by design, the cpu-budget spent on fitness evaluations of every new population does not change. However, an additional budget is re-

[2] This is true for smaller subset sizes (10-75%). For bigger subset sizes the distribution of model fitness deviates from the normal, and can be modeled as a Weibull distribution

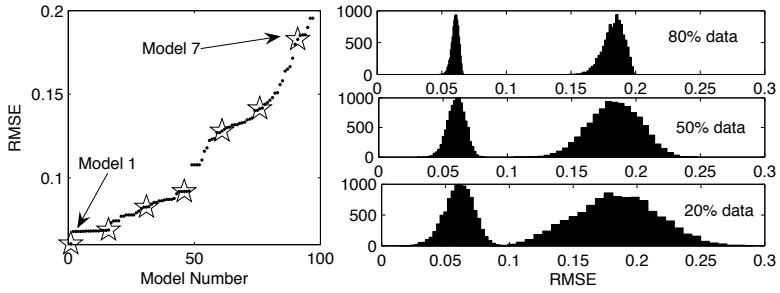


Figure 1-2. The left plot illustrates the sorted fitness values of an archive of a Kotanchek run at the last generation. Seven models (emphasized as stars) are selected from 100 archive individuals for further analysis. The plot at the right size shows the histograms of fitness distributions of model 1 and model 7 (the best and the worst from the selected set of models), computed on 10000 random subsets of 20,50,and 80% of the training data. We see that the widths of the histograms, i.e. the deviation of the estimated fitness from the true fitness, are inversely proportional to the size of the subset, and to the quality of the individual itself. E.g., model 1, which is the best one in the archive, has the most narrow fitness distribution, which becomes narrower as the subset size increases.

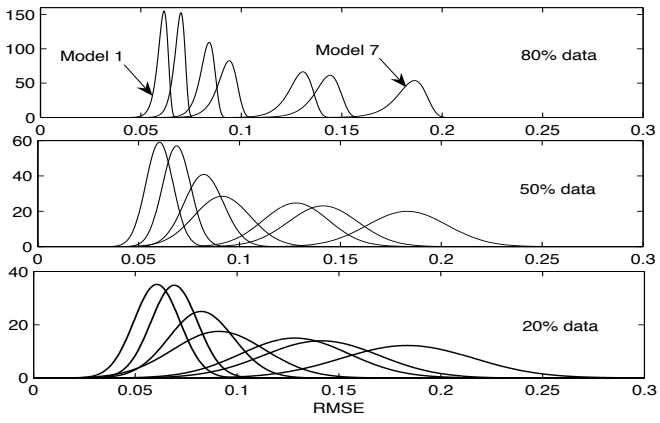


Figure 1-3. Approximated fitness distributions of the seven archive models from Figure 1-2 computed using random subsets of 20,50,and 80% of the original data set for the Kotanchek problem.

quired due to presence of an archive in ParetoGP. At each generation we create a new archive by merging the old one of the previous generation with the new population and selecting a fixed number of individuals located at the Pareto front in fitness-complexity space.

Since we randomly select a *new* subset to evaluate the population at every generation, in principle, the fitness of the archive needs to be reevaluated using the exact same subset. We considered this necessary, to make sure that we compare apples with apples, when updating the archive for the next step. At each generation the extra computational effort is equal to the product of the archive size and the number of records in the current subset. This implies that the additional effort grows when the subset size goes up during the run. For the case where the subset size increases from 10% to 100%, the archive re-evaluation step causes an increase of the total reference cpu-budget by 45%.

The budget of 145% can be called 'about the same' effort, but the increase is certainly not negligible. This makes us wonder whether we can reduce archive re-evaluations and still get solutions of a similar quality.

Soft ordinal ParetoGP - better solutions with less effort?

Why cpu time can be saved. Given the additional effort required to re-evaluate the archive at every new generation, we can question whether we really need to evaluate models on the same subset to get a good-enough ranking.

To analyse the influence on the ranking of two individuals we have to examine another distribution which is derived from a consideration of the difference in the estimated fitness of both individuals. This second distribution gives a direct estimate of the probability that you will have a wrong estimate of the rank of the two given individuals.

Below, we model the distributions of two different cardinal approaches to estimate the ranking of two individuals - taking the difference of fitness estimates obtained on the same subset, and taking the difference of fitness estimates obtained using different subsets of a similar size (see Figure 1-4).

The results of this analysis are unexpected. Comparing apples with apples gives us the most accurate estimates of the true difference in fitness, and correspondingly in the true ranking of the individuals. Surprisingly, the error in comparing apples to oranges is still small enough to make reasonably accurate decisions on the ranking of individuals even for small subset sizes. This statement seems to be general - we compared pairs of models of similar and different quality, sampled at various stages of the evolution, for various test problems.

Does more exploration lead to yet better solutions?

We assume that the essential elements of creating robust solutions are:

Table 1-2. Parameter settings of reference ParetoGP runs. Note, that the other experiments described in Cases 1-3 will have the same settings except for the population size, and the data set used for fitness evaluations.

Number of independent runs	30
Number of cascades	10
Number of generations per cascade	25
Total number of generations	250 (500 in PGPA)
Population size	100 (200 in PGPB)
Archive size	100
Run Performance measure	Area% under Pareto front
Accuracy measure	$1 - NMSE$ (smaller values preferred)
Complexity measure	Expressional
Population tournament size	5
Archive tournament size	3
Crossover rate	0.95
Mutation rate	0.05
Rate of mutation on terminals	0.3
Function set 1(*kotanchek, tower*)	$+$,$-$,$*$,$/$,e^x,e^{-x}, x^{real}, $x + real$,$x \cdot real$
Function set 2 (*maarten*)	Function set 1 $\bigcup \sin x, \cos x$

- **Abundant exploration**, caused by the use of larger population sizes and a softer selection process;
- **Sufficient exploitation**, caused by selecting potentially good parent models for further propagation. Despite the fact that the fidelity of the selection process is lower due to partial fitness evaluation, the exploitation of good models in the archive is still sufficient, due to the fact that winning models need to survive multiple low-fidelity screens to stay in the game and keep propagating. That said, goal softening by means of partial fitness evaluations introduces new modes of exploitation. It allows bad-but-lucky models to propagate short term, but guarantees that truly good models (which are good on all data points) will survive multiple screens and keep propagating long term.

Since abundant exploration combined with sufficient exploitation is essential for creating robust solutions, why should we insist on having a constant budget per generation? This can be achieved easily by allocating more cpu-budget to the initial part of the run (for exploration) at the expense of budget allocated to the end of run (for exploitation). This will be discussed further in the section on simulation results, Case 3.

Summary. Goal softening comes from:

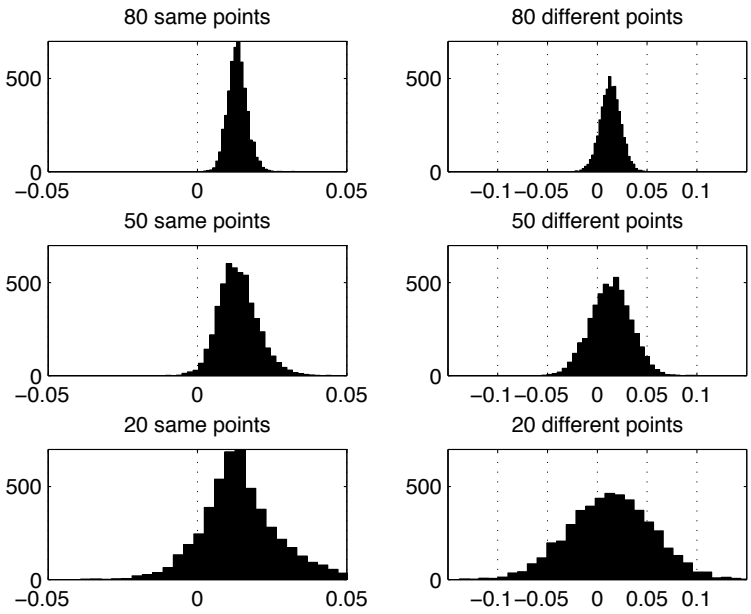


Figure 1-4. The histograms represent the difference in fitness values of the two archive models, calculated 10000 times on the same subset (the left-hand side), and different subsets of the same size (the right-hand side). The models used in this analysis are model 5 and model 6, selected in Figure 1-2. Notice, that the distributions on the left-hand side are much narrower than the ones on the right-hand side. This indicates, that a higher variance in the actual value of the difference in fitness values of the two models is introduced by evaluating them on different subsets. However, not the actual value of this difference is important. Only the sign matters to make a decision about the ranking. Thus, the fraction of the histogram area to the left of zero can be used to derive the probability of making an incorrect ranking. Notice, that even in the worst case (the down-right plot) an error in ranking is made in only 32% of evaluations. This plot corresponds to evaluations on 20% of the original data - an evaluation scheme used only at the beginning of the evolution. Given our commitment to soften the selection goals at the first generation, and only gradually improve the screening towards the end of the evolution - a 30-40% chance to make a mistake in ranking becomes more than appropriate.

- the use of subsets instead of the entire data set;
- use of *random* subsets of a given size at every generation; (has a regularizing effect on the models by reducing the chances for over-fitting)
- not re-evaluating the archive models at every generation, and giving them a chance to stay in the archive for a while;
- spending more cpu budget at the beginning of the evolution and less at the end.

There is a synergy of all these principles on guiding the search to better and more robust solutions faster...

5. Results

Experiment Setup

We selected three test problems for our experiments:

- *Maarten* (1.4) problem is drawn from model (1.4) and contains 101 records, with inputs sampled uniformly from the range $[0, 10]$,

$$f(x) = x^3 \exp^{-x} \cos x \sin x (\sin^2 x \cos x - 1); \tag{1.4}$$

- *Kotanchek* problem, drawn from (1.5), consists of 100 records, with inputs sampled randomly uniformly from the box $[0, 4] \times [0, 4]$,

$$f(x_1, x_2) = \frac{e^{-(x_2-1)^2}}{1.2 + (x_1 - 2.5)^2}; \tag{1.5}$$

- *Tower* problem is an industrial data set of a gas chromatography measurement of the composition of a distillation tower. The underlying data set contains 5000 records with noise and 23 potential input variables.

The ParetoGP parameters that we used in all experiments unless indicated otherwise are shown in Table 1-2.

The overall aim of all experiments was to get the highest quality results with the largest reproducibility and the lowest computational cost. In this chapter we focus on three case studies:

CASE 1. The first experiment setup is an extension of a setup that we discussed in an earlier paper (Smits and Vladislavleva, 2006). It is based on partial fitness evaluation of GP individuals on a *random* subset of the original data set. In this chapter the selection of subsets is uniformly random within the collection of the available data records. During the run we gradually increase the

Table 1-3. **Simulation Results for three test problems** The table represents the results of different experiments in terms of the median and the interquartile range of our performance measure over 50 independent runs and the normalized total number of function evaluations per run (cpu budget). The performance measure we use is the percentage of the area under the Pareto front of the archive solutions in complexity vs. fitness space. For all three quality characteristics - median and the IQR of the Pareto front area percentage, and the cpu budget - smaller values are preferred. There are three reference runs: PGP with the population size 100, and 250 generations, PGPA with twice the number of generations, and PGPB with twice the population size of the PGP runs.

Experiment	**Maarten**		**Kotanchek**		**Tower**		BUDGET
	$\tilde{\mu}$	IQR	$\tilde{\mu}$	IQR	$\tilde{\mu}$	IQR	per run
Reference							
PGP	1.81	1.39	2.20	0.735	1.55	0.325	250000
PGPA	1.23	0.93	2.05	0.786	1.39	0.334	500000
PGPB	1.57	1.12	2.06	0.630	1.41	0.334	500000
Case 1							
OPGP10	1.04	0.26	1.86	0.662	1.33	0.209	367169
OPGP20	1.02	0.34	2.02	0.633	1.37	0.368	372486
OPGP40	1.15	0.34	2.10	0.764	1.39	0.300	387703
OPGP60	1.24	0.54	2.18	0.831	1.57	0.362	404590
OPGP80	1.34	0.71	2.30	1.002	1.46	0.422	422122
Case 2							
Q10	1.05	0.31	1.93	0.431	1.32	0.250	273119
Q20	1.20	0.46	1.95	0.382	1.34	0.253	269886
Q40	1.24	0.65	2.18	0.652	1.41	0.285	268003
Q60	1.20	0.80	2.23	0.630	1.56	0.405	267790
Q80	1.76	1.89	2.14	0.710	1.44	0.380	268222
Case 3							
ER10	1.03	0.30	2.00	0.624	1.52	0.387	276610
ER20	1.10	0.29	1.92	0.571	1.35	0.296	271850
ER40	1.19	0.60	2.00	0.527	1.35	0.333	268876
ER60	1.42	1.14	2.12	0.520	1.39	0.287	268201
ER80	1.57	1.32	2.14	0.809	1.42	0.331	268379
ER100	1.65	1.75	2.32	0.802	1.64	0.327	250000

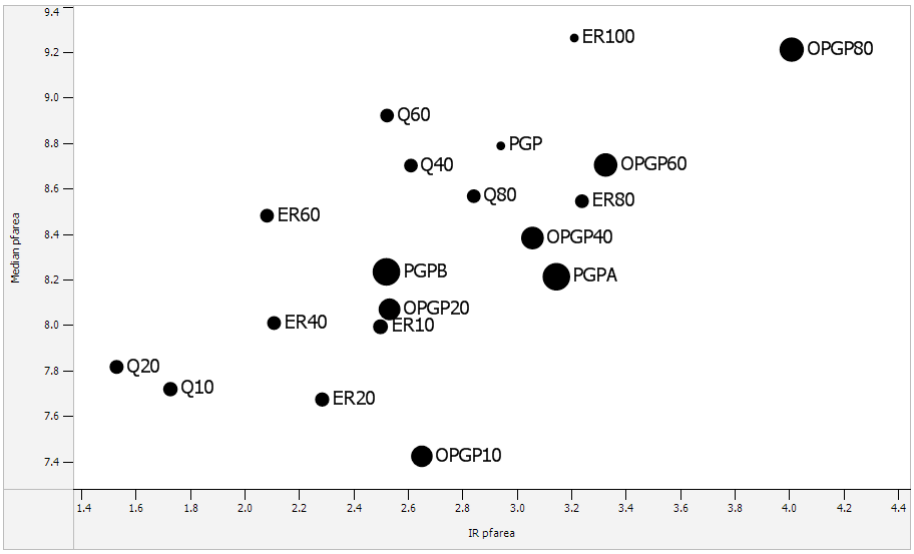
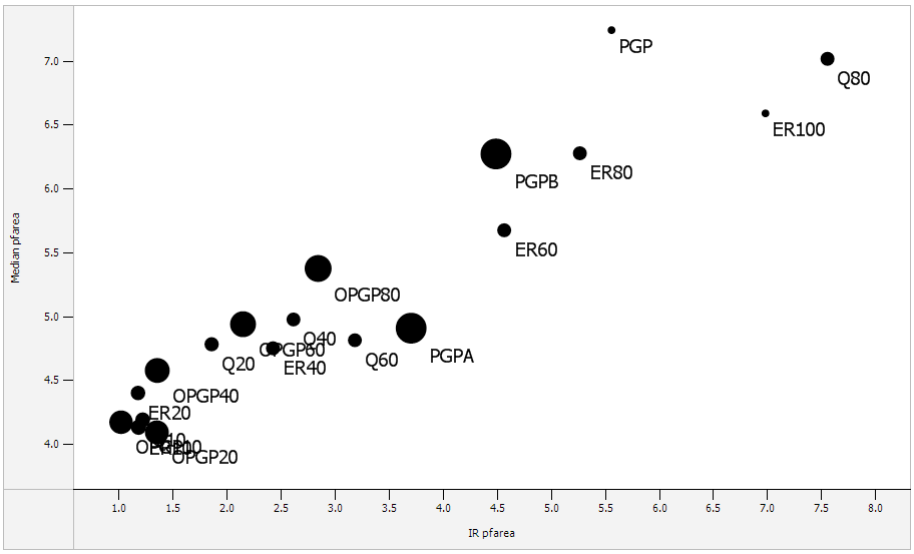


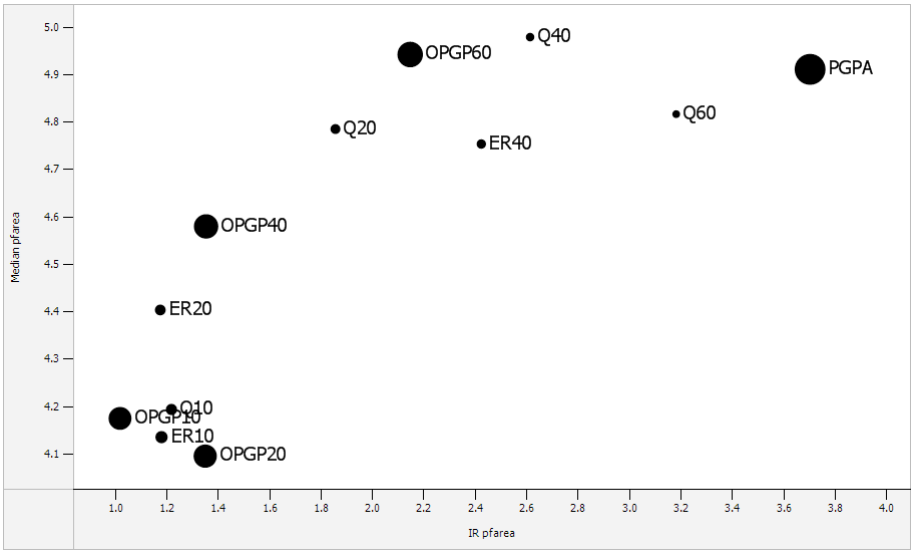
Figure 1-5. **Summary of Results for the Kotanchek problem.** The bubbles represent the results of the different experiments in Cases 1-3 in terms of the median performance measure over 50 independent runs (y-axis), interquartile range of the performance measure (x-axis), and the normalized total number of function evaluations per run (the size of the bubbles). We use the same labels as in Table 1-3. PGP - reference ParetoGP runs, OPGP - ordinal ParetoGP runs of Case 1, Q - quick ordinal runs of Case 2 with less frequent archive re-evaluations, ER - exploratory runs of Case 3 with re-allocated cpu-budget.The numbers in the labels indicate the starting level of the subset size. The area of interest is the lower-left corner of the graph. Note, that the most successful experiments are Q20,Q10, ER20, OPGP10, which confirms that starting from small subsets leads to solutions superior in terms of best median fitness and consistency of independent runs.

subset size while decreasing the population size. By doing so we pursue better exploration at the beginning of the evolution by evaluating more individuals at a time, albeit more coarsely, and gradually improve exploitation by refining the evaluations by adding more data points. Five different experiments were performed where the subset size at generation one was selected from a set of $\{10\%, 20\%, 40\%, 60\%, 80\%\}$ of the original data.

This time we want to keep the number of function evaluations per generation constant. Therefore, when the scheme of increasing the data subset size is chosen, and the number of records used for each generation is defined, the population size for a given generation is obtained by dividing the number of function evaluations (the budget) per generation by the number of records. All experiments were repeated more than 50 times to get reliable statistics.



(a) All experiments



(b) Magnified area of interest

Figure 1-6. **Summary of Results for the Maarten problem.** The bubbles represent the results of the different experiments in Cases 1-3 in terms of the median performance measure over 50 independent runs (y-axis), interquartile range of the performance measure (x-axis), and the normalized total number of function evaluations per run (the size of the bubbles). We use the same labels as in Table 1-3. PGP - reference ParetoGP runs, OPGP - ordinal ParetoGP runs of Case 1, Q - quick ordinal runs of Case 2 with less frequent archive re-evaluations, ER - exploratory runs of Case 3 with re-allocated cpu-budget. The numbers in the labels indicate the starting level of the subset size. The area of interest is the lower-left corner of the graph. The most successful experiments for the Maarten problem are OPGP10, Q10, ER10, OPGP20, which again confirms that starting from small subsets leads to solutions superior in terms of best median fitness and consistency of independent runs. Note, the substantial budget savings in Q10 and ER10, which correspond to Case 2, and Case 3 respectively.

CASE 2. In a second set of experiments we are repeating the set up described in Case 1 with one small modification. We are no longer re-evaluating individuals in the archive at every generation , but only at every 10-th generation. The cpu-budget assigned to these archive re-evaluations is cut by 90% compared with Case 1. This introduces further softening of the selection process, since archive individuals that accidentally obtain (unrealistically) high fitness values, can still survive in the archive for up to ten generations, and, hence, compete and propagate their features to their offspring. As noted in the previous section, the error in ranking caused by this approach are relatively small, and should still produce solutions comparable with the solutions of Case 1.

CASE 3. In the previous two cases the cpu-budget per generation was kept constant. In the third set of experiments we examine the effect of re-distributing the total cpu-budget in such a way, that we spend 50% more function evaluations at the first generation, gradually decrease the budget over the run, and end up with spending 50% less at the last generation, compared to the corresponding runs of Case 1 and 2. The total budget per run, however, does not change. The intention is to have even more exploration at the beginning of the run at the expense of exploitation at the end of the run. The scheme for archive re-evaluations is the same as in Case 2 - once every ten generations.

Simulation Results

The simulation results for the three cases are summarized in Table 1-3. We used the following criteria for estimating the performance of different experiments: the median and the interquartile range of the percentage of the area under the Pareto front for 50 independent replicates, and the total number of function evaluations per run normalized by the size of the training data set. For each criterion lower values indicate better performance.

The results from Table 1-3 are also displayed as bubble charts in Figures 1-5,1-6, and 1-7. The size of the bubbles in these charts is proportional to the budget that was spent for a particular experiment.

When we examine the results for **Case 1**, we observe a clear improvement of the median fitness as well as the IQR as we start with smaller and smaller subsets. The surprising fact is that even for very small datasets, like Maarten and Kotanchek, the optimal strategy is to start with a low subset size of 10 to 20%. This is most probably related to the low dimensionality of these problems. Also note, that the runs starting with the smaller subset sizes also consume less cpu-budget because the archive reevaluations are cheaper. The ordinal runs starting with small subset sizes (OPGP10 and OPGP20) considerably outperform the reference ParetoGP runs with a constant population size of 100 (PGP), and even those with twice the number of generations (PGPA) or twice the population size (PGPB).

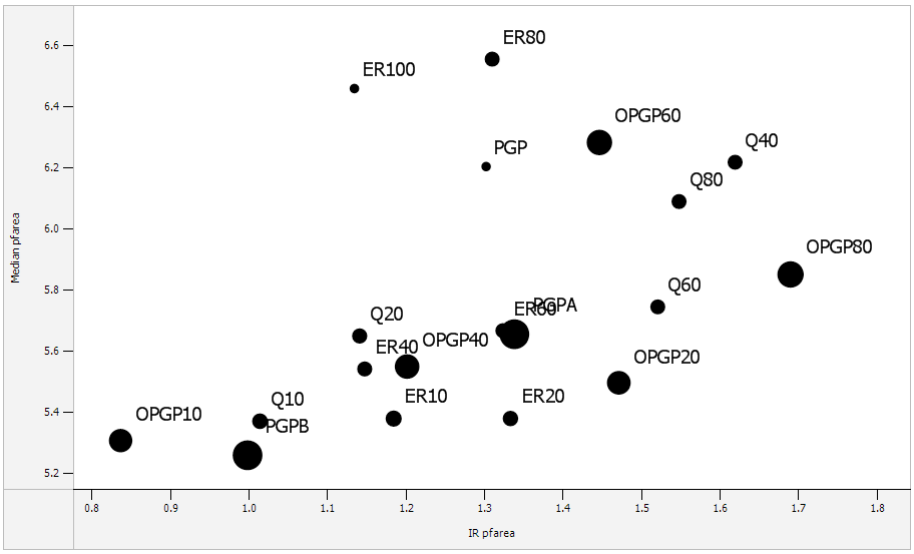


Figure 1-7. **Summary of Results for the Tower problem.** The most successful experiments for the Tower problem are OPGP10, Q10, and PGPB. The first two are no surprise, we observe the same results for other test problems. The PGPB corresponds to the reference ParetoGP runs that uses twice the population size for the same number of generations. The high quality of these results is caused by the nature of the tower problem. This real industrial problem is more challenging with respect to the variable selection (23 candidate inputs). Besides, the exact underlying input - output relationship may not exist. It is no surprise that PGPB runs with a bigger population size , and, hence, better exploration, produce better solutions than exploitative PGPA runs with more generations. It is important to note, that this improvement requires a higher cpu-budget.

In **Case 2**, we examine the effect of not reevaluating the archive individuals at every generation but only once every ten generations. The bubble charts show only a slight deterioration in the results for the median fitness, and the IQR, compared to the ordinal runs from Case 1. Considering the budget, these experiments are nevertheless very competitive. The advantage is that the cpu-budget now is actually very close to the original budget of the PGP reference run.

In **Case 3** the aim was to examine whether there was any benefit in relocating a part of the cpu-budget from the end to the beginning of a run. The results are comparable with the results of Case 2, however there is a clear deterioration in reproducibility of the independent replicates. We can speculate, that by relocating 25% of the cpu-budget from the second half of the run into the beginning, we actually assign too much weight to exploration and not enough to exploitation. While a thorough analysis of other budget distribution schemes is

required, the provisional conclusion is that a constant cpu-budget per generation is quite a good strategy to be used as a default.

6. Conclusions

We confirmed and refined our earlier findings that the use of goal softening consistently generates results that are superior both in terms of median fitness, consistency between independent runs, and required computational budget. We conclude that starting with subset sizes of only 10 to 20% generates optimal results for all three test problems.

We also confirmed that there is no need to reevaluate the archive individuals at every generation, keeping essentially the same quality of results at the lower cpu-budget. Finally we showed that keeping the number of function evaluations per generation constant over the run, is a good default setting and seems to provide the necessary balance between exploration and exploitation in soft ordinal ParetoGP.

Acknowledgment

The authors would like to thank Arthur Kordon for providing them with the data set corresponding to the tower problem. The authors would also like to thank Dick Den Hertog from Tilburg University for many discussions on ordinal optimization and evolutionary computation.

References

Gathercole, Chris and Ross, Peter (1994). Dynamic training subset selection for supervised learning in genetic programming. In Davidor, Yuval, Schwefel, Hans-Paul, and Männer, Reinhard, editors, *Parallel Problem Solving from Nature III*, volume 866 of *LNCS*, pages 312–321, Jerusalem. Springer-Verlag.

Ho, Y.-C., Cassandras, C.G., Chen, C.-H, and Dai, L (2000). Ordinal optimization and simulation. *Journal of the Operational Research Society*, 51:490–500.

Ho, Yu-Chi (2000). Soft optimization for hard problems, computerized lecture via private communication/distribution.

Keijzer, Maarten and Foster, James (2007). Crossover bias in genetic programming. In Ebner, Marc, O'Neill, Michael, Ekárt, Anikó, Vanneschi, Leonardo, and Esparcia-Alcázar, Anna Isabel, editors, *Proceedings of the 10th European Conference on Genetic Programming*, volume 4445 of *Lecture Notes in Computer Science*, Valencia, Spain. Springer.

Kotanchek, Mark, Smits, Guido, and Vladislavleva, Ekaterina (2006). Pursuing the pareto paradigm tournaments, algorithm variations & ordinal optimization. In Riolo, Rick L., Soule, Terence, and Worzel, Bill, editors, *Genetic*

Programming Theory and Practice IV, volume 5 of *Genetic and Evolutionary Computation*, chapter 3, pages –. Springer, Ann Arbor.

Langdon, W. B. and Poli, Riccardo (2002). *Foundations of Genetic Programming*. Springer-Verlag.

Lau, T.W. Edward and Ho, Yu-Chi (1997). Universal alignment probabilities and subset selection for ordinal optimization. *J. Optim. Theory Appl.*, 93(3):455–489.

Laumanns, Marco, Thiele, Lothar, Zitzler, Eckart, and Deb, Kalyanmoy (2002). Archiving with guaranteed convergence and diversity in multi-objective optimization. In *GECCO*, pages 439–447.

Smits, Guido, Kordon, Arthur, Vladislavleva, Katherine, Jordaan, Elsa, and Kotanchek, Mark (2005). Variable selection in industrial datasets using pareto genetic programming. In Yu, Tina, Riolo, Rick L., and Worzel, Bill, editors, *Genetic Programming Theory and Practice III*, volume 9 of *Genetic Programming*, chapter 6, pages 79–92. Springer, Ann Arbor.

Smits, Guido and Kotanchek, Mark (2004). Pareto-front exploitation in symbolic regression. In O'Reilly, Una-May, Yu, Tina, Riolo, Rick L., and Worzel, Bill, editors, *Genetic Programming Theory and Practice II*, chapter 17, pages 283–299. Springer, Ann Arbor.

Smits, Guido and Vladislavleva, Ekaterina (2006). Ordinal pareto genetic programming. In Yen, Gary G., Wang, Lipo, Bonissone, Piero, and Lucas, Simon M., editors, *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 3114 – 3120, Vancouver. IEEE Press.

Teller, Astro and Andre, David (1997). Automatically choosing the number of fitness cases: The rational allocation of trials. In Koza, John R., Deb, Kalyanmoy, Dorigo, Marco, Fogel, David B., Garzon, Max, Iba, Hitoshi, and Riolo, Rick L., editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 321–328, Stanford University, CA, USA. Morgan Kaufmann.

Zhang, Byoung-Tak and Cho, Dong-Yeon (1998). Genetic programming with active data selection. In McKay, R. I. Bob, Yao, X., Newton, Charles S., Kim, J.-H., and Furuhashi, T., editors, *Simulated Evolution and Learning: Second Asia-Pacific Conference on Simulated Evolution and Learning, SEAL'98. Selected Papers*, volume 1585 of *LNAI*, pages 146–153, Australian Defence Force Academy, Canberra, Australia. Springer-Verlag. published in 1999.

Zitzler, Eckart and Thiele, Lothar (1998). Multiobjective optimization using evolutionary algorithms - a comparative case study. In Eiben, A. E., Bäck, Thomas, Schoenauer, Marc, and Schwefel, Hans-Paul, editors, *PPSN*, volume 1498 of *Lecture Notes in Computer Science*, pages 292–304. Springer.